Package Object

PDS Management Council Technical Session

July 31, 2000

S. Joy and J. Wilf

Purpose

- The primary reason to collect files into packages is to circumvent existing standards (PDS labeling, ISO file naming).

- The current need is to be able to distribute software collections on archive volumes.
  - Software components do not necessarily comply with ISO file naming conventions.
  - Requiring PDS labels for each component of a large software collection is unreasonable.

Goal

- Develop a mechanism for describing a collection of files that have been “packaged” into a single file within a PDS label.

- Not attempt to address the logical package issue at this time. Address the need to be able to describe and distribute software packages collected into a single file quickly.

- Develop a standard that does not preclude future development work on a package object that can be generalized to describe both physical and logical file collections.

Requirements

- A single package file (physical) must be created by using tar to collect multiple files into a single file.
- A PDS label must be created to describe the package file.
  - The label must also be able to describe package elements that have the same root name as the package that are present outside of the package (executables, user’s guide, etc.) .
- The logical or unpackaged contents must be described.
  - Each file in the package needs to be identified, and file attributes:
    - file_name (with path included) or path_name and file_name
    - interchange_format
    - record_type
    - file type (source, executable, object library, document, test data, etc)
    - bytes
    - creation_date

    should be provided.
  - This attribute set is sufficient to allow the automated generation of PDS minimum labels for each file in the package.

Why TAR?

- ZIP and other file packaging techniques typically compress the original files making them inaccessible should the software not be supported in the future.
- TAR is only commonly available format that provides a packaging capability that does not modify the files within the package.
  - TAR merely concatenates files together adding header information in front of each file.
  - ASCII files remain viewable within the tar file.
  - With a bit of effort, the individual files within a tar file could be recreated using most common file editors.
- Using TAR avoids the potential problem of files not being accessible if the software is not supported in the future.

Solution

- Use the FILE object as a class (similar to DOCUMENT)
  - Explicitly point to all files with the same root name as the label and include a FILE object for each (no implicit file object).
  - Encourage use of the INTERCHAGE_FORMAT keyword by adding it to the optional keyword set.
  - Encourage use of the DESCRIPTION keyword by adding it to the optional keyword set.
  - Create "special instance" of FILE object for PACKAGE_FILE, similar to the SERIES and SPECTRUM forms of TABLE.
    - Discourage use of packaging for groups of files other than software.
    - Require a "MANIFEST_TABLE" object within the object. The manifest table specifies the attributes of each file in the package.
    - Require an ENCODING_TYPE keyword.
    - Encourage use of keywords SOFTWARE_NAME, PLATFORM, SOFTWARE_VERSION_ID, TECHNICAL_SUPPORT_TYPE by including them in the optional keyword set for the PACKAGE_FILE object.
  - Create a MANIFEST_TABLE object as a special instance of the TABLE object.
    - Require specific COLUMN objects (list provided previously)
    - Allow additional column objects as needed.

Example

- In this example, there are two versions of the software that will be included on an archive volume (Windows and Solaris). Both the executable file and the readme file are provided both within the software package and outside of the package file. Files are packaged into a single tar file for each supported platform.
- The SOFTWARE directory would look like:

```
+[SOFTWARE]
     |
     + SOFTINFO.TXT
     |
     +[PCWIN]
     |    + READMOC.EXE
     |    + READMOC.LBL
     |    + READMOC.TAB
     |    + READMOC.TAR
     |    + READMOC.TXT
     |
     +[SOLARIS]
     |    + READMOC.EXE
     |    + READMOC.LBL
     |    + READMOC.TAB
     |    + READMOC.TAR
     |    + READMOC.TXT
```

Example Label

```
PDS_VERSION_ID                    = 3
LABEL_REVISION_NOTE               = “E. Eliason, 1999-03-17;”
^PACKAGE_FILE                     = “READMOC.TAR”
^PACKAGE_MANIFEST_FILE            = “READMOC.TAB”
^EXECUTABLE_SOFTWARE_FILE         = “READMOC.EXE”
DESCRIPTION                       = “READMOC is a software package provided by the MOC team to decompress
     MOC images. The complete package, including source code, make files, etc.  is provided as a single TAR file
     (READMOC.TAR). A Microsoft Windows (NT, 95, 98) version of the executable file (READMOC.EXE) is also
     provided outside of the software package for direct usage on the supported platforms.”
OBJECT                            = EXECUTABLE_SOFTWARE_FILE
  INTERCHANGE_FORMAT              = BINARY
  RECORD_TYPE                     = UNDEFINED
  DESCRIPTION                     = “READMOC program executable to decompress MOC images.”
END_OBJECT                        = EXECUTABLE_SOFTWARE_FILE
OBJECT                            = PACKAGE_FILE
  INTERCHANGE_FORMAT              = BINARY
  RECORD_TYPE                     = UNDEFINED
  ENCODING_TYPE                   = TAR
  SOFTWARE_NAME                   = “READMOC”
  SOFTWARE_VERSION_ID             = “2.17”
  SOFTWARE_LICENCE_TYPE           = “PUBLIC DOMAIN”
  TECHNICAL_SUPPORT_TYPE          = “NONE”
  PLATFORM                        = {“WINDOWS NT”, “WINDOWS 95”, “WINDOWS 98”}
  DESCRIPTION                     = “Complete set of files for the READMOC software to decompress MOC images.”
  MANIFEST_TABLE                  = “READMOC.TAB”
END_OBJECT                        = PACKAGE_FILE
OBJECT                            = PACKAGE_MANIFEST_FILE
  RECORD_TYPE                     = “FIXED LENGTH”
  RECORD_BYTES                    = 100
  FILE_RECORDS                    = 15
  DESCRIPTION                     = “Table of contents for the READMOC software package tar file.”
  OBJECT                          = MANIFEST_TABLE
  END_ OBJECT                     = MANIFEST_TABLE
END_OBJECT                        = PACKAGE_MANIFEST_FILE
END
```

Example Manifest Table Object

```
OBJECT                          = MANIFEST_TABLE
   INTERCHANGE_FORMAT   = ASCII
   ROWS                         = 15
   COLUMNS                      = 5
   ROW_BYTES                    = 120
   OBJECT                       = COLUMN
     NAME                       = FILE_NAME
     DATA_TYPE                  = CHARACTER
     START_BYTE                 =1
     BYTES                      = 64
     DESCRIPTION                = “Full path and file name (POSIX standard) to the given file.”
   END_OBJECT                   = COLUMN
   OBJECT                       = COLUMN
     NAME                       = INTERCHANGE_FORMAT
     DATA_TYPE                  = CHARACTER
     START_BYTE                 = 66
     BYTES                      = 8
     DESCRIPTION                = “Interchange format of the named file (ASCII or BINARY).”
   END_OBJECT                   = COLUMN
   OBJECT                       = COLUMN
     NAME                       = “RECORD TYPE”
     DATA_TYPE                  = CHARACTER
     START_BYTE                 = 86
     BYTES                      = 10
     DESCRIPTION                = “Record type of the named  file (typically UNDEFINED or STREAM).”
   END_OBJECT                   = COLUMN
   OBJECT                       = COLUMN
     NAME                       = “FILE BYTES”
     DATA_TYPE                  = “ASCII INTEGER”
     START_BYTE                 = 98
     BYTES                      = 8
     DESCRIPTION                = “Size of the named file in bytes.”
   END_OBJECT                   = COLUMN
   OBJECT                       = COLUMN
     NAME                       = “CREATION DATE”
     DATA_TYPE                  = “CHARACTER”
     START_BYTE                 = 108
     BYTES                      = 10
     DESCRIPTION                = “File creation date.”
   END_OBJECT                   = COLUMN
END_OBJECT                      = TABLE
```

Summary

- Multiple files can be packaged into a single file by using tar. A single PDS label is required to describe the package. Tar is preferred because it allows the package contents to be recovered if the tar software is not supported in the future.
- Explicit FILE objects are used within the label to describe a package file and its associated files.
  – Files within the package can also exist outside the package as long as they are described by a PDS label (either within the package label or by using attached labels).
- The contents of a package are described in a manifest table. The manifest table contains sufficient information to generate FILE objects for each file contained in the package.
- Special instances of the FILE and TABLE objects are proposed to describe PACKAGE_FILEs and MANIFEST_TABLEs.
- Package file support requires no inherently new objects and few changes to the existing standards.